

[illegible]

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```



```
1 0001 0 MODULE showio (IDENT='V04-000',
2 0002 0 ADDRESSING MODE (EXTERNAL=GENERAL,
3 0003 0 NONEXTERNAL=LONG_RELATIVE)
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1
32 0032 1 ++
33 0033 1
34 0034 1 FACILITY: SHOW utility
35 0035 1
36 0036 1 ABSTRACT:
37 0037 1 This module contains the I/O processing routines.
38 0038 1
39 0039 1 ENVIRONMENT:
40 0040 1 VAX native, user mode.
41 0041 1
42 0042 1 AUTHOR: Gerry Smith CREATION DATE: 25-Aug-1983
43 0043 1
44 0044 1 MODIFIED BY:
45 0045 1
46 0046 1 V03-003 MCN0176 Maria del C. Nasr 11-Jul-1984
47 0047 1 Suppress output when /NOOUTPUT is specified.
48 0048 1
49 0049 1 V03-002 GAS0181 Gerry Smith 19-Sep-1983
50 0050 1 Add an alternate path in OPEN_OUTPUT, to allow JCP
51 0051 1 to display journals using the SHODEVxxx routines.
52 0052 1
53 0053 1 V03-001 GAS0174 Gerry Smith 25-Aug-1983
54 0054 1 Split SHOWMAIN into SHOWMAIN and SHOWIO. Also added
55 0055 1 the /OUTPUT qualifier.
56 0056 1
57 0057 1 --
```

SHOWIO
V04-000

J 14
16-Sep-1984 01:20:22 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:09:36 [CLIUTL.SRC]SHOWIO.B32;1

Page 2
(2)

: 59
: 60

0058 1 LIBRARY 'SYSS\$LIBRARY:STARLET';
0059 1 REQUIRE 'SRC\$:SHOWDEF';

: VAX/VMS common definitions
: SHOW common definitions

SHOWIO
V04-000

K 14
16-Sep-1984 01:20:22 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:09:36 [CLIUTL.SRC]SHOWIO.B32;1

Page 3
(3)

```
: 62      0158 1 FORWARD ROUTINE
: 63      0159 1      open_output : NOVALUE,      ! Open output file
: 64      0160 1      show$write_line : NOVALUE,    ! Write a line to output file
: 65      0161 1      show$print_line : NOVALUE,    ! Print an already-formatted line
: 66      0162 1      file_error : NOVALUE;         ! Signal an I/O error
: 67      0163 1
: 68      0164 1 EXTERNAL ROUTINE
: 69      0165 1      cli$get_value,                ! Get qualifier value
: 70      0166 1      cli$present;                  ! Test if qualifier present
: 71      0167 1
: 72      0168 1 OWN
: 73      0169 1      out_rab : $BLOCK[rab$c_bln],    ! Output RAB
: 74      0170 1      out_rss : $BLOCK[nam$c_maxrss], ! Resultant name string
: 75      0171 1      output_desc : $BLOCK[dsc$c_s_b[n]; ! Output file descriptor
: 76      0172 1
```

```
78 0173 1 GLOBAL ROUTINE open_output (isi) : NOVALUE =
79 0174 2 BEGIN
80 0175
81 0176 |---
82 0177 |
83 0178 |   Open output file
84 0179 |
85 0180 |   Inputs:
86 0181 |       isi - if called from JCP, then ISI is the internal stream identifier
87 0182 |             of an already-opened file.
88 0183 |---
89 0184 |
90 0185
91 0186 BUILTIN
92 0187     actualcount;
93 0188
94 0189 LOCAL
95 0190     desc : $BLOCK[dsc$s_bln],           ! Output descriptor
96 0191     out_fab : $BLOCK[fab$b_bln],       ! Output FAB
97 0192     out_nam : $BLOCK[nam$b_bln],       ! Output NAM block
98 0193     out_ess : $BLOCK[nam$b_maxrss];    ! Expanded string
99 0194
100 0195 |
101 0196 |   Check to see if from SHOW, or from JCP.  If there is an actual parameter
102 0197 |   passed, then this has been called from JCP, and all that is required is
103 0198 |   to dummy-up the RAB.
104 0199 |
105 0200 |   IF actualcount() NEQ 0                ! If from JCP,
106 0201 |   THEN                                ! then file already opened...
107 0202 |   BEGIN
108 0203 |       $RAB_INIT(RAB = out_rab);        ! Set up the RAB, and then
109 0204 |       out_fab[rab$b_wisi] = .isi;      ! stuff the ISI in place.
110 0205 |       RETURN;
111 0206 |   END;
112 0207 |
113 0208 |
114 0209 |   Fill in the RMS control blocks.
115 0210 |
116 0211 |   $FAB_INIT(FAB=out_fab,              ! Initialize output FAB
117 0212 |             DNM = 'SHOW.LIS',         ! Default to show.lis
118 0213 |             NAM=out_nam,              ! Point to a name block
119 0214 |             FOP=<ofp,sqo>,
120 0215 |             RAT=cr,
121 0216 |             FAC=put);
122 0217 |
123 0218 |
124 0219 |   Get the output file spec, if any.
125 0220 |
126 0221 |   IF cli$present(%ASCII 'OUTPUT')      ! If /OUTPUT, then
127 0222 |   THEN                                ! get whatever
128 0223 |   BEGIN                                ! the user typed in.
129 0224 |       $init_dyndesc(desc);             ! (If just /OUTPUT,
130 0225 |       cli$get_value(%ASCII 'OUTPUT',   ! default will be
131 0226 |                     desc);            ! SHOW.LIS)
132 0227 |       out_fab[fab$l_fna] = .desc[dsc$a_pointer];
133 0228 |       out_fab[fab$b_fns] = .desc[dsc$w_length];
134 0229 |   END
```



```
135 0230 2 ELSE
136 0231 2 ! Otherwise use NL: to suppress all output
137 0232 2 !
138 0233 2 BEGIN
139 0234 2 out_fab[fab$l_fna] = UPLIT BYTE('NL:');
140 0235 2 out_fab[fab$b_fns] = %CHARCOUNT('NL:');
141 0236 2 END;
142 0237 2
143 P 0238 2 $NAM_INIT(NAM=out_nam, ! Initialize NAM block
144 P 0239 2 RSS=nam$c_maxrss, ! Get a resultant name
145 P 0240 2 RSA=out_rss,
146 P 0241 2 ESS=nam$c_maxrss, ! Get an expanded name
147 0242 2 ESA=out_ess);
148 0243 2
149 P 0244 2 $RAB_INIT(RAB=out_rab, ! Initialize output RAB
150 0245 2 FAB=out_fab);
151 0246 2
152 0247 2 !
153 0248 2 ! Create the output file and connect record stream.
154 0249 2 !
155 0250 2 IF NOT $CREATE(FAB=out_fab)
156 0251 2 THEN file_error(show$_openout,
157 0252 2 out_fab,
158 0253 2 .out_fab[fab$l_sts],
159 0254 2 .out_fab[fab$l_stv]);
160 0255 2
161 0256 2 IF NOT $CONNECT(RAB=out_rab)
162 0257 2 THEN file_error(show$_openout,
163 0258 2 out_rab,
164 0259 2 .out_rab[rab$l_sts],
165 0260 2 .out_rab[rab$l_stv]);
166 0261 2
167 0262 2 !
168 0263 2 ! Point the OUTPUT_DESC to the resultant name string.
169 0264 2 !
170 0265 2 output_desc[dsc$w_length] = .out_nam[nam$b_rsl];
171 0266 2 output_desc[dsc$a_pointer] = out_rss;
172 0267 2
173 0268 2 RETURN;
174 0269 1 END;
```

```
.TITLE SHOWIO
.IDENT \V04-000\
.PSECT $PLITS,NOWRT,NOEXE,2
53 49 4C 2E 57 4F 48 53 00000 P.AAA: .ASCII \SHOW.LIS\
00 00 54 55 50 54 55 4F 00008 P.AAC: .ASCII \OUTPUT\<0><0>
010E0006 00010 P.AAB: .LONG 17694726
00000000' 00014 .ADDRESS P.AAC
00 00 54 55 50 54 55 4F 00018 P.AAE: .ASCII \OUTPUT\<0><0>
010E0006 00020 P.AAD: .LONG 17694726
00000000' 00024 .ADDRESS P.AAE
3A 4C 4E 00028 P.AAF: .ASCII \NL:\
.PSECT $OWNS,NOEXE,2
```

00000 OUT_RAB:.BLKB 68
00044 OUT_RSS:.BLKB 255
00143 .BLKB 1
00144 OUTPUT_DESC:
.BLKB 8

\$RMS_PTR= OUT_RAB
\$RMS_PTR= OUT_RAB
.EXTRN CLISGET_VALUE, CLISPRESENT
.EXTRN SYSSCREATE, SYSSCONNECT

.PSECT \$CODE\$,NOWRT,2

				01FC 00000	.ENTRY	OPEN_OUTPUT, Save R2,R3,R4,R5,R6,R7,R8	0173
		58	00000000V	EF 9E 00002	MOVAB	FILE_ERROR, R8	
		57	00000000V	EF 9E 00009	MOVAB	P.AAA, R7	
		56	00000000V	EF 9E 00010	MOVAB	\$RMS_PTR, R6	
		5E	FE48	CE 9E 00017	MOVAB	-440(SP), SP	
				6C 95 0001C	TSTB	(AP)	0200
				13 13 0001E	BEQL	1\$	
0044	8F	00		00 2C 00020	MOVCS	#0, (SP), #0, #68, \$RMS_PTR	0203
				66 00027			
		66	4401	8F B0 00028	MOVW	#17409, \$RMS_PTR	
		02	A6 04	AC B0 0002D	MOVW	ISI, OUT_RAB+2	0204
				04 00032	RET		0202
0050	8F	00		00 2C 00033 1\$:	MOVCS	#0, (SP), #0, #80, \$RMS_PTR	0216
			A8	AD 0003A			
		A8	AD 5003	8F B0 0003C	MOVW	#20483, \$RMS_PTR	
		AC	AD 20000040	8F D0 00042	MOVL	#536870976, \$RMS_PTR+4	
		BE	AD	01 90 0004A	MOVB	#1, \$RMS_PTR+22	
		C6	AD 0202	8F B0 0004E	MOVW	#514, \$RMS_PTR+30	
		D0	AD FF48	CD 9E 00054	MOVAB	OUT_NAM, \$RMS_PTR+40	
		D8	AD	67 9E 0005A	MOVAB	P.AAA, \$RMS_PTR+48	
		DD	AD	08 90 0005E	MOVW	#8, \$RMS_PTR+53	
			10	A7 9F 00062	PUSHAB	P.AAB	0221
		00000000G	00	01 FB 00065	CALLS	#1, CLISPRESENT	
			24	50 E9 0006C	BLBC	R0, 2\$	
		F8	AD 020E0000	8F D0 0006F	MOVL	#34471936, DESC	0224
			FC	AD D4 00077	CLRL	DESC+4	
			F8	AD 9F 0007A	PUSHAB	DESC	0225
			20	A7 9F 0007D	PUSHAB	P.AAD	
		00000000G	00	02 FB 00080	CALLS	#2, CLISGET_VALUE	
		D4	AD FC	AD D0 00087	MOVL	DESC+4, OUT_FAB+44	0227
		DC	AD F8	AD 90 0008C	MOVB	DESC, OUT_FAB+52	0228
				09 11 00091	BRB	3\$	0221
		D4	AD 28	A7 9E 00093 2\$:	MOVAB	P.AAF, OUT_FAB+44	0234
		DC	AD	03 90 00098	MOVB	#3, OUT_FAB+52	0235
0060	8F	00		00 2C 0009C 3\$:	MOVCS	#0, (SP), #0, #96, \$RMS_PTR	0242
			FF48	CD 000A3			
			6002	8F B0 000A6	MOVW	#24578, \$RMS_PTR	
		FF4A	CD	01 8E 000AD	MNEGB	#1, \$RMS_PTR+2	
		FF4C	CD 44	A6 9E 000B2	MOVAB	OUT_RSS, \$RMS_PTR+4	
		FF52	CD	01 8E 000B8	MNEGB	#1, \$RMS_PTR+10	
		FF54	CD	6E 9E 000BD	MOVAB	OUT_ESS, \$RMS_PTR+12	
0044	8F	00		00 2C 000C2	MOVCS	#0, (SP), #0, #68, \$RMS_PTR	0245
				66 000C9			

	66	4401	8F	B0	000CA	MOVW	#17409, \$RMS_PTR	
3C	A6	A8	AD	9E	000CF	MOVAB	OUT_FAB, \$RMS_PTR+60	
		A8	AD	9F	000D4	PUSHAB	OUT_FAB	0250
00000000G	00		01	FB	000D7	CALLS	#1, SYSS\$CREATE	
	10		50	E8	000DE	BLBS	R0, 4\$	
	7E	B0	AD	7D	000E1	MOVQ	OUT_FAB+8, -(SP)	0253
		A8	AD	9F	000E5	PUSHAB	OUT_FAB	0251
		007810A2	8F	DD	000E8	PUSHL	#788578	
	68		04	FB	000EE	CALLS	#4, FILE_ERROR	
			56	DD	000F1	PUSHL	R6	0256
00000000G	00		01	FB	000F3	CALLS	#1, SYSS\$CONNECT	
	10		50	E8	000FA	BLBS	R0, 5\$	
	7E	08	A6	7D	000FD	MOVQ	OUT_RAB+8, -(SP)	0259
		A8	AD	9F	00101	PUSHAB	OUT_FAB	0257
		007810A2	8F	DD	00104	PUSHL	#788578	
	68		04	FB	0010A	CALLS	#4, FILE_ERROR	
0144	C6	FF4B	CD	9B	0010D	MOVZBW	OUT_NAM+3, OUTPUT_DESC	0265
0148	C6	44	A6	9E	00114	MOVAB	OUT_RSS, OUTPUT_DESC+4	0266
			04	0011A		RET		0269

; Routine Size: 283 bytes, Routine Base: \$CODE\$ + 0000


```

176 0270 1 GLOBAL ROUTINE show$write_line : NOVALUE =
177 0271 2 BEGIN
178 0272 2
179 0273 2 ---
180 0274 2
181 0275 2 Write a record to the output file
182 0276 2
183 0277 2 Inputs:
184 0278 2
185 0279 2 CONTROL_STRING - address of an descriptor for a control string for $FAOL
186 0280 2 PARAM_LIST - address of the parameter list for input to $FAOL
187 0281 2
188 0282 2 ---
189 0283 2
190 0284 2 BUILTIN
191 0285 2 actualcount,
192 0286 2 actualparameter;
193 0287 2
194 0288 2 LOCAL
195 0289 2 dummy;
196 0290 2
197 0291 2 INCR dummy FROM 1 TO ACTUALCOUNT()/2 DO
198 0292 2 BEGIN
199 0293 2 LOCAL
200 0294 2 status,
201 0295 2 control_string,
202 0296 2 param_list,
203 0297 2 buffer : VECTOR[1024,BYTE],
204 0298 2 desc : VECTOR[2];
205 0299 2
206 0300 2 control_string = ACTUALPARAMETER(2*(.dummy-1) + 1);
207 0301 2 param_list = ACTUALPARAMETER(2*(.dummy-1) + 2);
208 0302 2
209 0303 2 desc[0] = %ALLOCATION(buffer);
210 0304 2 desc[1] = buffer;
211 0305 2
212 P 0306 4 IF NOT (status = $FAOL(CTRSTR = .control_string,
213 P 0307 4 OUTBUF = desc,
214 P 0308 4 OUTLEN = desc,
215 0309 4 PRMLST = .param_list))
216 0310 3 THEN SIGNAL_STOP(.status);
217 0311 3
218 0312 3 out_rab[rab$w_rsz] = .desc[0];
219 0313 3 out_rab[rab$l_rbf] = .desc[1];
220 0314 4 IF NOT $PUT(RAB=out_rab)
221 0315 3 THEN SIGNAL(show$writeerr,
222 0316 3 1, output_desc,
223 0317 3 .out_rab[rab$l_sts],
224 0318 3 .out_rab[rab$l_stv]);
225 0319 2 END;
226 0320 2
227 0321 2 RETURN;
228 0322 1 END;

```

.EXTRN SYSS\$FAOL, SYSS\$PUT

			001C	00000	.ENTRY	SHOW\$WRITE_LINE, Save R2,R3,R4	: 0270
	54	00000000'	EF	9E	MOVAB	OUT_RAB+34, R4	:
	5E	FBF8	CE	9E	MOVAB	-1032(SP), SP	:
	53		6C	9A	MOVZBL	(AP), R3	: 0291
	53		02	C6	DIVL2	#2, R3	:
			52	D4	CLRL	DUMMY	:
			64	11	BRB	3\$	:
50	52		01	78	ASHL	#1, DUMMY, R0	: 0300
	51	FC	AC40	D0	MOVL	-4(AP)[R0], CONTROL_STRING	:
50	52		01	78	ASHL	#1, DUMMY, R0	: 0301
	50		6C40	D0	MOVL	(AP)[R0], PARAM_LIST	:
	6E	0400	8F	3C	MOVZWL	#1024, DESC	: 0303
	04	AE	08	AE	MOVAB	BUFFER, DESC+4	: 0304
			50	DD	PUSHL	PARAM_LIST	: 0309
			04	AE	PUSHAB	DESC	:
			08	AE	PUSHAB	DESC	:
			51	DD	PUSHL	CONTROL_STRING	:
	00000000G	00	04	FB	CALLS	#4, SYS\$FAOL	:
		09	50	E8	BLBS	STATUS, 2\$	:
			50	DD	PUSHL	STATUS	: 0310
	00000000G	00	01	FB	CALLS	#1, LIB\$STOP	:
		64	6E	B0	MOVW	DESC, OUT_RAB+34	: 0312
	06	A4	04	AE	MOVL	DESC+4, OUT_RAB+40	: 0313
			DE	A4	PUSHAB	OUT_RAB	: 0314
	00000000G	00	01	FB	CALLS	#1, SYS\$PUT	:
		17	50	E8	BLBS	R0, 3\$	:
		7E	E6	A4	MOVQ	OUT_RAB+8, -(SP)	: 0317
			0122	C4	PUSHAB	OUTPUT_DESC	: 0315
			01	DD	PUSHL	#1	:
		007810D2	8F	DD	PUSHL	#7868626	:
	00000000G	00	05	FB	CALLS	#5, LIB\$SIGNAL	:
98		52	53	F3	AOBLEQ	R3, DUMMY, 1\$	: 0291
			04	00080	RET		: 0322

; Routine Size: 129 bytes, Routine Base: \$CODE\$ + 011B


```
230 0323 1 GLOBAL ROUTINE show$print_line (length, buffer) : NOVALUE =
231 0324 2 BEGIN
232 0325 2
233 0326 2 ---
234 0327 2
235 0328 2 Write a record to the output file. This routine differs from
236 0329 2 SHOW$WRITE_LINE in that what is passed is the address and length
237 0330 2 of a buffer, ready to print.
238 0331 2
239 0332 2 Inputs:
240 0333 2     LENGTH - length of buffer
241 0334 2     BUFFER - address of buffer.
242 0335 2
243 0336 2 ---
244 0337 2
245 0338 2 out_rab[rab$w_rsz] = .length;
246 0339 2 out_rab[rab$l_rbf] = .buffer;
247 0340 2 IF NOT $PUT(RAB=out_rab)
248 0341 2 THEN SIGNAL(show$writeerr,
249 0342 2     1, output_desc,
250 0343 2     .out_rab[rab$l_sts],
251 0344 2     .out_rab[rab$l_stv]);
252 0345 2
253 0346 2 RETURN;
254 0347 1 END;
```

			0004 00000	.ENTRY	SHOW\$PRINT_LINE, Save R2	: 0323
	52	00000000'	EF 9E 00002	MOVAB	OUT_RAB+34, R2	: 0338
	62	04	AC B0 00009	MOVW	LENGTH, OUT_RAB+34	: 0339
06	A2	08	AC D0 0000D	MOVL	BUFFER, OUT_RAB+40	: 0340
		DE	A2 9F 00012	PUSHAB	OUT_RAB	: 0343
00000000G	00		01 FB 00015	CALLS	#1, SYSS\$PUT	: 0341
	17		50 E8 0001C	BLBS	R0, 1\$	:
	7E	E6	A2 7D 0001F	MOVQ	OUT_RAB+8, -(SP)	: 0347
		0122	C2 9F 00023	PUSHAB	OUTPUT_DESC	:
			01 DD 00027	PUSHL	#1	:
		007810D2	8F DD 00029	PUSHL	#7868626	:
00000000G	00		05 FB 0002F	CALLS	#5, LIB\$SIGNAL	:
			04 00036 1\$:	RET		: 0347

; Routine Size: 55 bytes, Routine Base: \$CODE\$ + 019C


```

256 0348 1 ROUTINE file_error(message, fab, sts, stv) : NOVALUE =
257 0349 BEGIN
258 0350
259 0351 ---
260 0352
261 0353 FUNCTIONAL DESCRIPTION:
262 0354
263 0355 This routine signals an error for a file.
264 0356
265 0357 Inputs:
266 0358
267 0359 message Message
268 0360 fab Address of the fab
269 0361 sts, stv STS and STV values
270 0362
271 0363 ---
272 0364
273 0365 MAP
274 0366 fab : REF $BBLOCK;
275 0367 BIND
276 0368 nam = .fab[fab$l_nam] : $BBLOCK;
277 0369 LOCAL
278 0370 filedesc : $BBLOCK[dsc$c_s_bln];
279 0371
280 0372
281 0373 CH$FILL(0, dsc$c_s_bln, filedesc);
282 0374
283 0375 IF .nam[nam$b_rsl] NEQ 0 ! If resultant name present
284 0376 THEN
285 0377 BEGIN
286 0378 filedesc[dsc$w_length] = .nam[nam$b_rsl];
287 0379 filedesc[dsc$a_pointer] = .nam[nam$[_rsa]];
288 0380 END
289 0381 ELSE IF .nam[nam$b_esl] NEQ 0 ! If expanded name present
290 0382 THEN
291 0383 BEGIN
292 0384 filedesc[dsc$w_length] = .nam[nam$b_esl];
293 0385 filedesc[dsc$a_pointer] = .nam[nam$[_esa]];
294 0386 END
295 0387 ELSE
296 0388 BEGIN
297 0389 filedesc[dsc$w_length] = .fab[fab$b_fns]; ! Use filename string
298 0390 filedesc[dsc$a_pointer] = .fab[fab$[_fna]]; ! if all else fails
299 0391 END;
300 0392
301 0393
302 0394 2 SIGNAL(.message, 1, filedesc, .sts, .stv);
303 0395 1 END;

```

```

                                00FC 0000 FILE_ERROR:
                                .WORD      Save R2,R3,R4,R5,R6,R7
5E                               SUBL2     #8, SP
57                               MOVL      FAB, R7
                                08 02 00002
                                08 AC D0 00005

```

```

: 0348
:
: 0368

```

SHOWIO
V04-000

G 15
16-Sep-1984 01:20:22 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:09:36 [CLIUTL.SRC]SHOWIO.B32;1

Page 12
(7)

08	00	56	28	A7	D0	00009	MOVL	40(R7), R6	:	
		6E		00	2C	0000D	MOVCS	#0, (SP), #0, #8, FILEDESC	:	0373
				6E		00012			:	
			03	A6	95	00013	TSTB	3(R6)	:	0375
				0B	13	00016	BEQL	1\$	:	
		6E	03	A6	9B	00018	MOVZBW	3(R6), FILEDESC	:	0378
04	AE		04	A6	D0	0001C	MOVL	4(R6), FILEDESC+4	:	0379
				19	11	00021	BRB	3\$	:	0375
			0B	A6	95	00023	TSTB	11(R6)	:	0381
				0B	13	00026	BEQL	2\$	:	
		6E	0B	A6	9B	00028	MOVZBW	11(R6), FILEDESC	:	0384
04	AE		0C	A6	D0	0002C	MOVL	12(R6), FILEDESC+4	:	0385
				09	11	00031	BRB	3\$	:	
		6E	34	A7	9B	00033	MOVZBW	52(R7), FILEDESC	:	0389
04	AE		2C	A7	D0	00037	MOVL	44(R7), FILEDESC+4	:	0390
	7E		0C	AC	7D	0003C	MGVQ	STS, -(SP)	:	0394
			08	AE	9F	00040	PUSHAB	FILEDESC	:	
				01	DD	00043	PUSHL	#1	:	
			04	AC	DD	00045	PUSHL	MESSAGE	:	
	00000000G	00		05	FB	00048	CALLS	#5, LIB\$SIGNAL	:	
				04	00	0004F	RET		:	0395

; Routine Size: 80 bytes, Routine Base: \$CODE\$ + 01D3

SHOWIO
V04-000

H 15
16-Sep-1984 01:20:22
14-Sep-1984 12:09:36

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]SHOWIO.B32;1

Page 13
(8)

: 305 0396 1 END
: 306 0397 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
\$OWNS	332	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$PLITS	43	NOVEC, NOWRT, RD, NOEXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)
\$CODES	547	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	106	1	581	00:01.0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:SHOWIO/OBJ=OBJ\$:SHOWIO MSRC\$:SHOWIO/UPDATE=(ENH\$:SHOWIO)

: Size: 547 code + 375 data bytes
: Run Time: 00:13.4
: Elapsed Time: 00:36.4
: Lines/CPU Min: 1774
: Lexemes/CPU-Min: 38311
: Memory Used: 135 pages
: Compilation Complete

0056 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

